**PLSQL**

Step 1: Create a Package Specification

CREATE OR REPLACE PACKAGE employee\_pkg AS

    -- Public variables

    g\_bonus\_rate NUMBER := 0.05;

    -- Public procedures and functions

    PROCEDURE update\_salary(p\_emp\_id IN NUMBER, p\_new\_salary IN NUMBER);

    FUNCTION get\_salary(p\_emp\_id IN NUMBER) RETURN NUMBER;

END employee\_pkg;

Step 2: Create a Package Body

CREATE OR REPLACE PACKAGE BODY employee\_pkg AS

    -- Private variables (not visible outside the package)

    g\_private\_variable NUMBER := 100;

    -- Public procedure implementation

    PROCEDURE update\_salary(p\_emp\_id IN NUMBER, p\_new\_salary IN NUMBER) IS

    BEGIN

        UPDATE employees

        SET salary = p\_new\_salary

        WHERE employee\_id = p\_emp\_id;

    END update\_salary;

    -- Public function implementation

    FUNCTION get\_salary(p\_emp\_id IN NUMBER) RETURN NUMBER IS

        v\_salary NUMBER;

    BEGIN

        SELECT salary

        INTO v\_salary

        FROM employees

        WHERE employee\_id = p\_emp\_id;

        RETURN v\_salary;

    END get\_salary;

    -- Private procedure (not declared in the specification)

    PROCEDURE private\_procedure IS

    BEGIN

        -- Implementation of the private procedure

    END private\_procedure;

END employee\_pkg;

BEGIN

    -- Gather statistics on the 'employees' table

    DBMS\_STATS.GATHER\_TABLE\_STATS(

        ownname    => 'YOUR\_SCHEMA', -- Replace with your schema name

        tabname    => 'EMPLOYEES',

        cascade    => TRUE           -- Gather statistics on the indexes as well

    );

    DBMS\_OUTPUT.PUT\_LINE('Statistics gathered successfully for the EMPLOYEES table.');

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('An error occurred while gathering statistics: ' || SQLERRM);

END;

user\_scheduler\_jobs

**Schedule Job:-**

BEGIN

    -- Create a scheduler job

    DBMS\_SCHEDULER.create\_job (

        job\_name        => 'GATHER\_STATS\_JOB',

        job\_type        => 'PLSQL\_BLOCK',

        job\_action      => 'BEGIN DBMS\_STATS.GATHER\_TABLE\_STATS(ownname => ''YOUR\_SCHEMA'', tabname => ''EMPLOYEES'', cascade => TRUE); END;',

        start\_date      => SYSTIMESTAMP,

        repeat\_interval => 'FREQ=DAILY; BYHOUR=0; BYMINUTE=0; BYSECOND=0',

        enabled         => TRUE

    );

    DBMS\_OUTPUT.PUT\_LINE('Scheduler job "GATHER\_STATS\_JOB" created and enabled successfully.');

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('An error occurred while creating the scheduler job: ' || SQLERRM);

END;

Enable Job DB-

-- Enable a job

BEGIN

    DBMS\_SCHEDULER.enable('GATHER\_STATS\_JOB');

    DBMS\_OUTPUT.PUT\_LINE('Scheduler job "GATHER\_STATS\_JOB" enabled.');

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('An error occurred while enabling the scheduler job: ' || SQLERRM);

END;

-- Disable a job

BEGIN

    DBMS\_SCHEDULER.disable('GATHER\_STATS\_JOB');

    DBMS\_OUTPUT.PUT\_LINE('Scheduler job "GATHER\_STATS\_JOB" disabled.');

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('An error occurred while disabling the scheduler job: ' || SQLERRM);

END;

Dropping Job:-

BEGIN

    DBMS\_SCHEDULER.drop\_job('GATHER\_STATS\_JOB');

    DBMS\_OUTPUT.PUT\_LINE('Scheduler job "GATHER\_STATS\_JOB" dropped.');

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('An error occurred while dropping the scheduler job: ' || SQLERRM);

END;

**%Type**

In PL/SQL, the %TYPE attribute is used to declare a variable that has the same data type as a column in a table or a field in a record. This attribute ensures that the variable automatically adopts the data type of the referenced column or field, which helps maintain consistency and reduces the risk of errors due to data type mismatches.

DECLARE

    v\_employee\_id employees.employee\_id%TYPE;

BEGIN

    SELECT employee\_id INTO v\_employee\_id

    FROM employees

    WHERE first\_name = 'John' AND last\_name = 'Doe';

    DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || v\_employee\_id);

END;

/

**%Rowtype**

The %ROWTYPE attribute in PL/SQL is used to declare a record variable that can store an entire row of data from a table or a cursor. This attribute provides a way to reference the structure of a table row or a cursor row dynamically, ensuring that the record variable automatically adopts the data types of the corresponding table columns or cursor fields.

DECLARE

    v\_employee employees%ROWTYPE;

BEGIN

    SELECT \* INTO v\_employee

    FROM employees

    WHERE employee\_id = 1;

    DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || v\_employee.employee\_id);

    DBMS\_OUTPUT.PUT\_LINE('First Name: ' || v\_employee.first\_name);

    DBMS\_OUTPUT.PUT\_LINE('Last Name: ' || v\_employee.last\_name);

    DBMS\_OUTPUT.PUT\_LINE('Email: ' || v\_employee.email);

    DBMS\_OUTPUT.PUT\_LINE('Hire Date: ' || v\_employee.hire\_date);

END;

/

CollectionType:-

You can declare a PL/SQL collection type that is based on the row type of an existing table using %ROWTYPE. This allows you to create an array-like collection that can hold multiple rows of the same structure as the table.

Below is an example demonstrating how to declare a PL/SQL collection type based on the asin\_dim table using %ROWTYPE and how to use it in a PL/SQL block:

  type   changeListType is table of asin\_dim%rowtype index by binary\_integer;

  insList changeListType;  --List of inserts

  updList changeListType;  --List of updates

  insRec asin\_dim%rowtype; --single insert record

  updRec asin\_dim%rowtype; --single update record

About **gather\_table\_stats**:-

DBMS\_STATS.GATHER\_TABLE\_STATS (

    ownname          IN   VARCHAR2,

    tabname          IN   VARCHAR2,

    partname         IN   VARCHAR2 DEFAULT NULL,

    estimate\_percent IN   NUMBER   DEFAULT to\_estimate\_percent\_type (DBMS\_STATS.AUTO\_SAMPLE\_SIZE),

    block\_sample     IN   BOOLEAN  DEFAULT FALSE,

    method\_opt       IN   VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE AUTO',

    degree           IN   NUMBER   DEFAULT to\_degree\_type (DBMS\_STATS.DEFAULT\_DEGREE),

    granularity      IN   VARCHAR2 DEFAULT 'DEFAULT',

    cascade          IN   BOOLEAN  DEFAULT TRUE,

    stattab          IN   VARCHAR2 DEFAULT NULL,

    statid           IN   VARCHAR2 DEFAULT NULL,

    statown          IN   VARCHAR2 DEFAULT NULL,

    no\_invalidate    IN   BOOLEAN  DEFAULT FALSE,

    force            IN   BOOLEAN  DEFAULT FALSE

 );

**Parameters:**

* **ownname:** The schema name where the table resides.
* **tabname:** The name of the table for which to gather statistics.
* **partname:** The name of the partition (if the table is partitioned).
* **estimate\_percent:** The percentage of rows to estimate (default is DBMS\_STATS.AUTO\_SAMPLE\_SIZE).
* **block\_sample:** Whether to use block sampling instead of row sampling.
* **method\_opt:** Options for gathering column statistics.
* **degree:** The degree of parallelism to use for gathering statistics.
* **granularity:** The level of granularity for gathering statistics (e.g., 'DEFAULT', 'ALL', 'GLOBAL', 'PARTITION').
* **cascade:** Whether to gather statistics on the table's indexes as well.
* **stattab, statid, statown:** Used for storing statistics in a user-specified table.
* **no\_invalidate:** Whether to invalidate dependent cursors.
* **force:** Whether to force gathering statistics even if they are locked.

If use-specified tables aren’t defined check on the below tables:-

* user\_tab\_statistics
* user\_tab\_columns
* user\_indexes
* user\_tab\_statistics

Declaring PLSQL Table:-

DECLARE

-- Declare a record type

TYPE employee\_record IS RECORD (

   emp\_id     NUMBER,

   emp\_name   VARCHAR2(100),

   emp\_salary NUMBER

);

-- Declare a table type of the record type

TYPE employee\_table IS TABLE OF employee\_record;

-- Declare a variable of the table type

employees employee\_table;

BEGIN

-- Initialize the table

employees := employee\_table();

-- Populate the table

employees.EXTEND(3); -- Extend the table to hold 3 elements

employees(1) := employee\_record(1, 'John Doe', 50000);

employees(2) := employee\_record(2, 'Jane Smith', 60000);

employees(3) := employee\_record(3, 'Bob Johnson', 70000);

-- Example of accessing the table elements

FOR i IN employees.FIRST..employees.LAST LOOP

   DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || employees(i).emp\_id);

   DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || employees(i).emp\_name);

   DBMS\_OUTPUT.PUT\_LINE('Employee Salary: ' || employees(i).emp\_salary);

END LOOP;

END;

/

**Pragma AUTONOMOUS\_TRANSACTION:-**

The PRAGMA AUTONOMOUS\_TRANSACTION directive in PL/SQL is used to mark a procedure, function, or trigger as an autonomous transaction. This means the block will execute independently of the calling transaction. Autonomous transactions allow you to perform SQL operations that commit or roll back without affecting the main transaction.

In below example log\_message is being declared as **PRAGMA AUTONOMOUS\_TRANSACTION**.This process is being called in other processes and can be commit independently without considering it’s parent processes.

CREATE OR REPLACE PROCEDURE log\_message(p\_message IN VARCHAR2) AS

   PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

   -- Insert the message into a logging table

   INSERT INTO log\_table (message, log\_date)

   VALUES (p\_message, SYSDATE);

   -- Commit the transaction

   COMMIT;

END log\_message;

**Locking record for update:-**

DECLARE

   -- Declare a variable to hold the customer ID

   v\_customer\_id NUMBER := 1;

BEGIN

   -- Lock the record for update

   FOR rec IN (

      SELECT customer\_id, first\_name, last\_name

      FROM customers

      WHERE customer\_id = v\_customer\_id

      FOR UPDATE

   ) LOOP

      -- Update the locked record

      UPDATE customers

      SET last\_name = 'Smith'

      WHERE customer\_id = rec.customer\_id;

      -- Optionally, you can perform other operations here

   END LOOP;

   -- Commit the transaction

   COMMIT;

EXCEPTION

   WHEN OTHERS THEN

      -- Roll back the transaction in case of an error

      ROLLBACK;

      RAISE;

END;

/

**Locking Table for update:-**

-- Wait for 5 mins to acquire the lock

SELECT customer\_id, first\_name, last\_name

FROM customers

WHERE customer\_id = v\_customer\_id

FOR UPDATE WAIT 5;  -- Wait up to 5 seconds for the lock

-- Fail if lock can't be obtained

SELECT customer\_id, first\_name, last\_name

FROM customers

WHERE customer\_id = v\_customer\_id

FOR UPDATE NOWAIT;  -- Fail immediately if the lock cannot be obtained

--Lock multiple tables:-

SELECT customer\_id, first\_name, last\_name

FROM customers

WHERE status = 'ACTIVE'

FOR UPDATE;

The **ALTER SESSION SET TRACEFILE\_IDENTIFIER** command in Oracle allows you to specify an identifier that will be included in the names of trace files generated for the current session. This is particularly useful for debugging and performance tuning, as it helps you distinguish between trace files created for different sessions or purposes.

ALTER SESSION SET TRACEFILE\_IDENTIFIER = 'identifier\_string';

Sample:-

--Set Tracefile Identifier:-

ALTER SESSION SET TRACEFILE\_IDENTIFIER = 'DEBUG\_SESSION\_1';

--Enable SQL Trace

EXEC DBMS\_SESSION.SET\_SQL\_TRACE(TRUE);

--Execute Your SQL Statements(Sample SQL statements for tracing)

SELECT \* FROM employees;

-- Disable SQL Trace

EXEC DBMS\_SESSION.SET\_SQL\_TRACE(FALSE);

--Check Trace FileThe trace files generated will include the identifier in their names,

making them easier to identify. For example,

if the default directory for trace files is used,

you might find a file named something like ora\_12345\_DEBUG\_SESSION\_1.trc.

SQL Profiler:-

In Oracle Database, the **Oracle SQL Profiler** is a tool used for analyzing the performance of SQL queries by providing detailed information about their execution. The profiler helps identify and diagnose performance issues by generating detailed execution reports.

### Key Aspects of Oracle SQL Profiler

1. **Purpose**:
   * To profile and analyze SQL queries to understand their performance characteristics.
   * To identify resource bottlenecks, long-running queries, and inefficient operations.
2. **Components**:
   * **Profiler Framework**: The underlying mechanism that collects performance data during SQL execution.
   * **Profiler Tables**: Internal tables where profiling data is stored.
   * **Profiler Views**: Views that provide access to profiling data.
3. **Usage**:
   * **Start Profiling**: Begin the profiling session for a specific SQL statement or session.
   * **Analyze Results**: Examine the profiling data to understand the execution characteristics of your SQL queries.

**ETL Profiler:-**

-- Start profiling

BEGIN

   DBMS\_PROFILER.START\_PROFILER(

      run\_comment    => 'ETL Process XYZ',

      run\_comment1   => 'Profiler initiated by etl\_control\_pkg'

   );

END;

/

-- Execute SQL statements to profile

SELECT \* FROM employees WHERE department\_id = 10;

-- Stop profiling

BEGIN

   DBMS\_PROFILER.STOP\_PROFILER;

END;

/

-- Query the profiling data

SELECT \*

FROM prof$run

WHERE run\_comment = 'ETL Process XYZ';

### Profiling Data Queries

1. **Query to Retrieve Profiling Run Information**

To get an overview of all profiling runs, including comments and other metadata:

SELECT run\_id, run\_comment, run\_comment1, run\_comment2, run\_comment3, run\_start\_time, run\_end\_time

FROM prof$run

ORDER BY run\_start\_time DESC;

1. **Query to Retrieve Detailed Profiling Information**

To get detailed profiling data for a specific run, including SQL statements and execution statistics:

SELECT r.run\_id, s.sql\_id, s.sql\_text, s.execution\_count, s.total\_elapsed\_time, s.total\_cpu\_time

FROM prof$statement s

JOIN prof$run r ON s.run\_id = r.run\_id

WHERE r.run\_id = 1001

ORDER BY s.total\_elapsed\_time DESC;

1. **Query to Retrieve Profiling Data for Execution of PL/SQL Blocks**

To get information about PL/SQL block executions:

SELECT r.run\_id, p.plsql\_id, p.plsql\_text, p.execution\_count, p.total\_elapsed\_time

FROM prof$plsql p

JOIN prof$run r ON p.run\_id = r.run\_id

WHERE r.run\_id = 1001

ORDER BY p.total\_elapsed\_time DESC;

### Interpretation of Output

* **RUN\_ID**: Unique identifier for the profiling run. Use this to correlate data across different views.
* **SQL\_ID**: Identifier for the SQL statement. This helps in identifying and correlating specific SQL queries.
* **SQL\_TEXT**: The actual SQL statement that was executed.
* **EXECUTION\_COUNT**: Number of times the SQL statement or PL/SQL block was executed.
* **TOTAL\_ELAPSED\_TIME**: Total time taken to execute the SQL statement or PL/SQL block. Helps in identifying long-running queries.
* **TOTAL\_CPU\_TIME**: Total CPU time consumed. Useful for understanding CPU-intensive operations.

### Troubleshooting Tips

* **Identify Long-Running Queries**: Look for SQL statements with high TOTAL\_ELAPSED\_TIME to identify performance bottlenecks.
* **Analyze CPU Usage**: High TOTAL\_CPU\_TIME indicates queries that are CPU-intensive and may benefit from optimization.
* **Check Execution Count**: High EXECUTION\_COUNT for certain queries or PL/SQL blocks may suggest frequent re-execution, which could be optimized.

By analyzing the profiling data using these queries, you can gain insights into the performance characteristics of your SQL queries and PL/SQL code, helping you to diagnose and resolve performance issues effectively.

Important Terms:-

* Flush
* getProcessedRecord
* Activesessions – Stats
* AttachedProcess
* SessionTraces
* SessionProfiler
* BeingProcess
  + Start the ETL prcess
  + Lock the ETL\_Process record from ETL\_Control table, you created for your own use.
  + No other Active sessions (Call ActiveSession function mentioned above)
  + Maintain History table (Maintain etl\_control\_hist table )
* UpdateProcess
* EndProcess
* Dblog

Key generation logic:-

select (select nvl(max(device\_dim\_key),1) from device\_dim) + rownum,

The /\*+ parallel(16) \*/ hint in Oracle SQL is used to instruct the database to execute the query using parallel execution with a specified number of parallel execution servers. This can improve the performance of large queries by distributing the workload across multiple CPU cores.

Here's how you might use this hint in a SQL query:

SELECT /\*+ parallel(16) \*/ COUNT(1)

FROM your\_table;

In PL/SQL, using an associative array (also known as an index-by table) to store a list of records can be very useful for batch processing or managing collections of rows in memory. Here’s how you can define and use such an associative array to handle a list of inserts into a table.

### Step-by-Step Implementation

#### 1. Define the Associative Array Type

First, you define a type for your associative array that matches the structure of the rows you want to work with. In this example, you’re using the %ROWTYPE attribute to define a record type based on the structure of the customer\_device\_dim table.

-- Define an associative array type for the list of inserts

DECLARE

    TYPE changeListType IS TABLE OF customer\_device\_dim%ROWTYPE INDEX BY BINARY\_INTEGER;

    insList changeListType;  -- List of inserts

BEGIN

    -- Example of populating the associative array and using it for inserts

    -- Here we assume the customer\_device\_dim table has columns id, device, and status

    -- Add some records to the associative array

    insList(1).id := 101;

    insList(1).device := 'Device A';

    insList(1).status := 'Active';

    insList(2).id := 102;

    insList(2).device := 'Device B';

    insList(2).status := 'Inactive';

    -- Loop through the associative array and insert records into the table

    FOR i IN 1 .. insList.COUNT LOOP

        INSERT INTO customer\_device\_dim (id, device, status)

        VALUES (insList(i).id, insList(i).device, insList(i).status);

    END LOOP;

    -- Commit the transaction to make changes persistent

    COMMIT;

END;

/

### In real life instead of value you

### Explanation:

1. **Type Definition**:
   * changeListType: Defines an associative array where each element is of the same type as a row in the customer\_device\_dim table.
   * insList: An instance of changeListType used to store records for batch processing.
2. **Populating the Array**:
   * You manually set values for each element in insList. Each element corresponds to a row you want to insert into the customer\_device\_dim table.
3. **Inserting Records**:
   * Loop through the associative array (insList) and insert each record into the customer\_device\_dim table.
4. **Commit**:
   * The COMMIT statement ensures that all changes made during the transaction are saved to the database.

DECLARE

    -- Define an associative array type

    TYPE changeListType IS TABLE OF customer\_device\_dim%ROWTYPE INDEX BY BINARY\_INTEGER;

    insList changeListType;  -- List of inserts

    -- Counter for the associative array

    counter INTEGER := 1;

BEGIN

    -- Select records into the associative array

    FOR rec IN (SELECT \* FROM some\_source\_table) LOOP

        -- Populate the associative array with the fetched rows

        insList(counter).id := rec.id;

        insList(counter).device := rec.device;

        insList(counter).status := rec.status;

        -- Increment the counter

        counter := counter + 1;

    END LOOP;

    -- Loop through the associative array and insert records into the table

    FOR i IN 1 .. insList.COUNT LOOP

        INSERT INTO customer\_device\_dim (id, device, status)

        VALUES (insList(i).id, insList(i).device, insList(i).status);

    END LOOP;

    -- Commit the transaction to make changes persistent

    COMMIT;

END;

/

The /\*+ APPEND \*/ hint in Oracle SQL is used to optimize INSERT operations by directly appending new rows to the end of a table, bypassing some of the standard mechanisms that might otherwise cause performance overhead. This hint can be particularly useful when you are performing bulk inserts and want to improve performance.

Here’s how you can use the /\*+ APPEND \*/ hint in the context of inserting data into a table.

Sample PLSQL Block:-

create or replace package etl\_marketplace\_dim

is

/\*

manages the data in the marketplace\_dim table

\*/

  /\*reads records from staging table(s) and loads into the marketplace\_dim.

  Return: the number of records inserted.

  \*/

  function load return number;

end etl\_marketplace\_dim;

/

create or replace package body etl\_marketplace\_dim

is

  savepoint\_mask varchar2(255) := 'YYYY-MM-DD HH24:MI:SS';

  insRec marketplace\_dim%rowtype; --single insert record

  updRec marketplace\_dim%rowtype; --single update record

  /\*-------------------------------------------------------------------

  load

  \-------------------------------------------------------------------\*/

  function load return number

  is

    etl etl\_control%rowtype;

    pragma autonomous\_transaction;

  begin

    --Start a new ETL session

    etl := etl\_control\_pkg.beginProcess('etl\_marketplace\_dim.load');

    --setup our savepoint key for the first load.

    if etl.savepoint\_key is null then

      etl.savepoint\_key := '2000-01-01 00:00:00';--'YYYY-MM-DD HH24:MI:SS'

    end if;

    --setup todo

    select count(1) into etl.todo

    from o\_ms\_marketplaces

    where dw\_last\_updated >= to\_date(etl.savepoint\_key,savepoint\_mask)

    ;

    --query the staged data and populate a record object of the target table

    for stageRec in (

      select

        marketplace\_id,

        case when upper(trim(default\_region)) = 'NA' then 1

             when upper(trim(default\_region)) = 'EU' then 2

             when upper(trim(default\_region)) = 'FE' then 3

             when upper(trim(default\_region)) = 'IN' then 4

             when upper(trim(default\_region)) = 'SA' then 5

             else 0

        end region\_id,

        marketplace\_name,

        default\_region default\_region\_code,

        case when upper(trim(default\_region)) = 'NA' then 'North America'

             when upper(trim(default\_region)) = 'EU' then 'Europe'

             when upper(trim(default\_region)) = 'FE' then 'Far East'

             when upper(trim(default\_region)) = 'IN' then 'India'

             when upper(trim(default\_region)) = 'SA' then 'South America'

             else 'UNKNOWN'

        end default\_region\_name,

        default\_language\_code,

        default\_currency\_code,

        '\_UC' coins\_currency\_code,

        default\_time\_zone\_name,

        mktplc\_domain\_name\_prefix,

        sysdate dw\_creation\_date,

        sysdate dw\_last\_updated,

        dw\_last\_updated savepoint\_key

      from o\_ms\_marketplaces

      where dw\_last\_updated >= to\_date(etl.savepoint\_key,savepoint\_mask)

      order by dw\_last\_updated

    ) loop

      --fail the load if we were not able to map something

      if stageRec.region\_id = 0 then

        raise\_application\_error(-2001,'unable to map a region id to region code "' || stageREc.default\_region\_code || '"');

      end if;

      if stageRec.default\_region\_name = 'UNKNOWN' then

        raise\_application\_error(-2001,'unable to map a region name to region code "' || stageREc.default\_region\_code || '"');

      end if;

      etl.done := etl.done + 1;

      etl.savepoint\_key := to\_char(stageRec.savepoint\_key,savepoint\_mask);

      --populate an insert record.

      insRec.marketplace\_id            := stageRec.marketplace\_id;

      insRec.region\_id                 := stageRec.region\_id;

      insRec.marketplace\_name          := stageRec.marketplace\_name;

      insRec.default\_region\_code       := stageRec.default\_region\_code;

      insRec.default\_region\_name       := stageRec.default\_region\_name;

      insRec.default\_language\_code     := stageRec.default\_language\_code;

      insRec.default\_currency\_code     := stageRec.default\_currency\_code;

      insRec.coins\_currency\_code       := stageRec.coins\_currency\_code;

      insRec.default\_time\_zone\_name    := stageRec.default\_time\_zone\_name;

      insRec.mktplc\_domain\_name\_prefix := stageRec.mktplc\_domain\_name\_prefix;

      insRec.dw\_creation\_date          := stageRec.dw\_creation\_date;

      insRec.dw\_last\_updated           := stageRec.dw\_last\_updated;

      --Check for existing record based on natural key

      begin

        select \*

        into updRec

        from marketplace\_dim md

        where insRec.marketplace\_id = md.marketplace\_id

        ;

        --found an existing dimension record, check for changes.

        if insRec.marketplace\_name          != updRec.marketplace\_name

        or insRec.default\_region\_code       != updRec.default\_region\_code

        or insRec.default\_region\_name       != updRec.default\_region\_name

        or insRec.default\_language\_code     != updRec.default\_language\_code

        or insRec.default\_currency\_code     != updRec.default\_currency\_code

        or insRec.coins\_currency\_code       != updRec.coins\_currency\_code

        or insRec.default\_time\_zone\_name    != updRec.default\_time\_zone\_name

        or insRec.mktplc\_domain\_name\_prefix != updRec.mktplc\_domain\_name\_prefix

        then

          --change detected.  this is fixed dimension - so just update the attribs

          update marketplace\_dim

          set  marketplace\_name = insRec.marketplace\_name,

               default\_region\_code = insRec.default\_region\_code,

               default\_region\_name = insRec.default\_region\_name,

               default\_language\_code = insRec.default\_language\_code,

               default\_currency\_code = insRec.default\_currency\_code,

               coins\_currency\_code = insRec.coins\_currency\_code,

               default\_time\_zone\_name = insRec.default\_time\_zone\_name,

               mktplc\_domain\_name\_prefix = insRec.mktplc\_domain\_name\_prefix,

               dw\_last\_updated = insRec.dw\_last\_updated

          where marketplace\_id = insRec.marketplace\_id;

          etl.updated := etl.updated + 1;

        end if;

      exception when no\_data\_found then

        insert into marketplace\_dim values insRec;

        etl.inserted := etl.inserted + 1;

      end;

    end loop;

    commit;

    --make sure the default record is there as well

    insert into marketplace\_dim

      select

      0,

      0,

      'UNKNOWN',

      '?',

      'UNKNOWN',

      '?',

      '?',

      '?',

      'UNKNOWN',

      'UNKNOWN',

      sysdate,

      sysdate

      from dual where not exists ( select 1 from marketplace\_dim where marketplace\_id = 0  )

    ;

    commit;

    etl.inserted := etl.inserted + sql%rowcount;

    etl.todo := etl.todo + sql%rowcount;

    etl.done := etl.done + sql%rowcount;

    if etl.errors > 0 then

      raise\_application\_error(-20001,etl.errors || ' errors were encountered during load.  See dblog\_table for error details');

    end if;

    --flush our ETL results

    etl\_control\_pkg.endProcess(etl);

    return etl.inserted;

  exception when others then

    etl.errors := etl.errors + 1;

    --flush our ETL results

    etl\_control\_pkg.endProcess(etl);

    dblog.error();

    raise;

  end load;

begin

  -- Initialization: none yet

  null;

end etl\_marketplace\_dim;

/

Sample ETL Blocks:-

function beginProcess(

    processName IN ETL\_CONTROL.process\_name%TYPE,

    parentETLProcess in varchar2 default null

  ) RETURN ETL\_CONTROL%ROWTYPE

  is

    PRAGMA AUTONOMOUS\_TRANSACTION;

    etl ETL\_CONTROL%ROWTYPE;

  begin

    --if a parent process was specified, make sure that it is an active etl process.

    if parentETLProcess is not null then

      if parentETLProcess = processName then

        raise\_application\_error(-20001,'Unable to start ETL process. Parent process name is the same as the process name');

      end if;

      if getProcessStatus( parentETLProcess ) != 'ACTIVE' then

      raise\_application\_error(-20001,

       'Unable to start ETL process.  Parent processes "' || parentETLProcess || '" status of "'  || getProcessStatus( parentETLProcess ) || '"  is not "ACTIVE"'

      );

      end if;

      childrenList( parentETLProcess ) := 'Y';

    end if;

    --get control record, and lock it so no one else can start this process at the same time

    begin

      select \*

      into etl

      from ETL\_CONTROL

      where process\_name = processName

      for update;

      --make sure there is not a session running this ETL

      assertActiveETLSession(etl);

      --add last control record to history

      insert into etl\_control\_hist select \* from etl\_control where process\_name = processName;

    exception when no\_data\_found then

      insert into ETL\_CONTROL (process\_name) values ( processName ) ;

      etl.process\_name := processName;

      etl.log\_level := 'INFO';

      etl.log\_hist\_days := 90;

      etl.profile\_run\_count := 0;

      etl.trace\_run\_count := 0;

    end;

    --make sure the logging control record matches whats in the etl control record.

    dblog.configure(etl.process\_name,etl.log\_level,etl.log\_hist\_days);

    --setup some defaults

    etl.Status := 'ACTIVE';

    etl.Output := null;

    etl.Begin\_Time := SYSDATE;

    etl.End\_Time := NULL;

    etl.Todo := NULL;

    etl.Done := 0;

    etl.elapsed := '0 00:00:00';

    etl.pct\_Done := 0;

    etl.Rps := 0;

    etl.Eta := NULL;

    etl.todo := 0;

    etl.done := 0;

    etl.inserted := 0;

    etl.updated := 0;

    etl.deleted := 0;

    etl.errors := 0;

    etl.warnings := 0;

    etl.parent\_process\_name := parentETLProcess;

    etl.last\_updated := sysdate;

    dblog.setLogger(etl.process\_name);

    dblog.info('process "' || etl.process\_name || '" is starting');

    --enable trace as required

    startSessionTrace(etl);

    --enable profiler as required

    startSessionProfiler(etl);

    --update session

    dbms\_application\_info.set\_module(module\_name => substr(etl.process\_name,1,48),action\_name => 'ETL Running');

    --save the record to the database

    previousStatusList(etl.process\_name) := etl;

    flush(etl);

    return etl;

  end beginProcess;

**SQL Written by Me:-**

##Evaluations

select \*

from andes.learn.evaluations

where

evaluation\_id = 'TTELERN202407011532503418d4c7';

-- Evaluation Details --

select

    ev.evaluation\_id,

    ev.evaluation\_name,

    ev.sections\_enabled,

    \*

from

    andes.learn.evaluations ev-- 123 only has unique evaluations available under each LMS.

where

    evaluation\_id = 'TTELERN202407011532503418d4c7'

    and

creation\_date = '2024-07-01T15:32:50.530381Z'

--creation\_date = (select max(creation\_date) from andes.learn.evaluations ev2 where ev.evaluation\_id = ev2.evaluation\_id) -- for latest evaluation

-- PK is eval\_id and creation\_date reason for adding max(creation\_date) is there are multiple records present for each evaluation\_id.

-- Additional information about evaluations attached to the trainings. Note that evaluation\_id here can be used while taking join with training\_id from training table.

select

    e.evaluation\_id as evaluation\_id,

    t.learn\_training\_id as learn\_training\_id,

    t.version as training\_version,

    t.training\_type as training\_type,

    eta.attachments\_enabled as attachments\_enabled,

    eta.max\_attempts as max\_attempts,

    eta.passing\_score\_percentage as passing\_score\_percentage,

    eta.number\_of\_questions as number\_of\_questions

from

    andes.learn.evaluations e

join andes.learn.training t on

    e.evaluation\_id = t.learn\_training\_id

    and TO\_CHAR(TO\_TIMESTAMP(e.creation\_date,'YYYY-MM-DD"T"HH24:MI:SS.FF6"Z"'), 'YYYY-MM-DD HH24:MI:SS.FF6') = TO\_CHAR(TO\_TIMESTAMP(t.creation\_date, 'YYYY-MM-DD HH24:MI:SS.FF3'), 'YYYY-MM-DD HH24:MI:SS.FF6')

join andes.learn.evaluation\_training\_attributes eta on

    t.learn\_training\_id = eta.learn\_training\_id

    and t.version = eta.version

where

    t.learn\_training\_id = 'TTELERN20240701205759ab765095'

    and t.version = '1.0';

-- Submission detail selecetd for evaluation. It will generate duplicates without evaluation creation\_date.

select

    s.user\_id as evaluation\_submitted\_by\_user,

    e.evaluation\_name,

    s.submission\_id,

    s.creation\_date as submission\_creation\_date,

    s.status as submission\_completion\_status,

    s.attempt\_number as submission\_attempt\_number,

    s.score as evaluation\_submission\_score,

    s.percentage\_score as percent\_evaluation\_submission\_score,

    s.\*

from

    andes.learn.evaluation\_submissions s

join andes.learn.evaluations e on

    e.evaluation\_id = s.evaluation\_id

where

    e.evaluation\_id = 'TTELERN202407011532503418d4c7'

    and

e.creation\_date = '2024-07-01T15:32:50.530381Z';

-- Evaluation submission for a specific user --

select

    s.user\_id as evaluation\_submitted\_by\_user,

    u.papi\_displayname as amzn\_employee\_name,

    case

        when u.papi\_bi\_jobstatus = 'A' then 'Active'

        when u.papi\_bi\_jobstatus = 'I' then 'Inactive'

        else 'Undefined'

    end as Empl\_Status,

    e.evaluation\_id as evaluation\_id,

    e.evaluation\_name,

    s.submission\_id,

    s.creation\_date as submission\_creation\_date,

    s.status as submission\_completion\_status,

    s.attempt\_number as submission\_attempt\_number,

    s.score as evaluation\_submission\_score,

    s.percentage\_score as percent\_evaluation\_submission\_score,

    s.\*

from

    andes.learn.evaluation\_submissions s

join andes.learn.evaluations e on

    e.evaluation\_id = s.evaluation\_id

join andes.learn.users u on

    u.id = s.user\_id

where

    e.evaluation\_id = 'TTELERN202407011532503418d4c7'

    and

e.creation\_date = '2024-07-01T15:32:50.530381Z'

    and s.user\_id = 'b8eafca2-e24f-4ee8-97da-b3665079ce90';

-- User Table details --

select

papi\_bi\_legacyemployeeid as amzn\_employee\_id,

papi\_displayname as amzn\_employee\_name,

papi\_bi\_jobstatus

from andes.learn.users

where id = '4672efae-5a51-4e67-9607-338fac259970' limit 10;

-- question, question\_type and correct\_answers

select

    e.evaluation\_id as evaluation\_id,

    e.evaluation\_name as evaluation\_name,

    q.question\_type as question\_type,

    t.text as question\_asked\_in\_eval,

    t.text\_type as question\_type,

    ct.text as correct\_answer,

    ct.text\_type as option\_type --,\*

from

    andes.learn.evaluations e

join andes.learn.evaluation\_form\_questions f on

    e.evaluation\_id = f.evaluation\_id and

    e.creation\_date = f.evaluation\_creation\_date

join andes.learn.evaluation\_questions q on

    q.question\_id = f.question\_id

join andes.learn.evaluation\_texts t on

    q.question\_id = t.text\_id

join andes.learn.evaluation\_texts ct on

    q.correct\_answer = ct.text\_id

where

    e.evaluation\_id = 'TTELERN202407011532503418d4c7'

    and

    e.creation\_date = '2024-07-01T15:32:50.530381Z'

-- Evaluation\_response provided in submission by user  --

-- answer here is given as an option number instead of answer\_id there is no way we can identify which answer is selecetd--

select

    s.submission\_id as submission\_id,

    q.question\_type as question\_type,

    t.text as question\_in\_submission,

    t.text\_type as question\_type,

    er.answer as answer\_given\_by\_user ,

    er.status as status\_of\_ans,

    er.score as score,

    u.papi\_displayname

from

    andes.learn.evaluation\_submissions s

join andes.learn.evaluation\_responses er on

    er.submission\_id = s.submission\_id and

    er.submission\_creation\_date = s.creation\_date

join andes.learn.evaluation\_questions q on

    q.question\_id = er.question\_id

join andes.learn.evaluation\_texts t on

    q.question\_id = t.text\_id

join andes.learn.evaluation\_texts ct on

    q.correct\_answer = ct.text\_id

join andes.learn.users u on

    u.id = s.user\_id

where

    s.submission\_id = 'cd6366a5-bc2b-532a-ad9f-15b08995700b'

ILT:-

select \* from andes.learn.training t

where t.learn\_training\_id = 'TPTLERN202407090021474204d7d9' limit 10;

select \* from andes.learn.ilt\_training\_attributes ita

where ita.learn\_training\_id = 'TPTLERN202407090021474204d7d9' limit 10;

-- ILT related information --

select

    t.learn\_training\_id as training\_id,

    t.version as training\_version,

    tt.training\_title,

    ita.venue\_type as venue\_type,

    ita.is\_mandatory as is\_mandatory,

    ita.registration\_max\_size as registration\_max\_size,

    ita.registration\_min\_size as registration\_min\_size,

    ita.registration\_deadline as registration\_deadline

from

    andes.learn.training t

join andes.learn.training\_title tt

on

    t.learn\_training\_id = tt.learn\_training\_id

    and t.version = tt.version

join

    andes.learn.ilt\_training\_attributes ita

on

    t.learn\_training\_id = ita.learn\_training\_id

    and

    t.version = ita.version

where

    t.learn\_training\_id = 'TPTLERN202407090021474204d7d9'

    and

tt.is\_default\_language = 'true';

------ ILT staff related information -----

with recursive numbers(n) as (

select

    1 as n

union all

select

    n + 1

from

    numbers

where

    n <= 1000

    -- Define the termination condition here

),

staff\_info as (

select

    ilts.learn\_training\_id as ilt\_learn\_training,

    ilts.version as version,

    ilts.staff\_type as ilt\_staff\_type,

    split\_part(ilts.staff\_user\_ids,

    ',',

    numbers.n) as ilt\_staff\_user\_id,

    split\_part(ilts.staff\_registration\_status,

    ',',

    numbers.n) as ilt\_staff\_registration\_status

from

    andes.learn.ilt\_staff ilts

join numbers on

    split\_part(ilts.staff\_user\_ids,

    ',',

    numbers.n) <> ''

--where

    --ilts.learn\_training\_id = 'TPTLERN20240702220824fd937171'

)

select

    si.ilt\_learn\_training as ilt\_training\_id,

    si.ilt\_staff\_type as ilt\_staff\_type,

    u.id as ilt\_user\_id,

    u.papi\_displayname as ilt\_staff\_display,

    u.papi\_bi\_jobstatus as ilt\_staff\_jobstatus,

    u.papi\_bi\_login as ilt\_staff\_bi\_login

from

    staff\_info si

left join andes.learn.users u on

    si.ilt\_staff\_user\_id = u.id;

Example:-

select  u.id, u.papi\_displayname, u.papi\_bi\_jobstatus, u.papi\_bi\_login  from

andes.learn.users u

where id in (

'46f5d0f2-6f77-4f9a-9692-457b8f1439c0' ,

'796fed1e-17ef-44f7-952f-13152060790c' ,

'2aa79c4d-af51-4a75-ba1d-77fd1cfd91ac' ,

'bf276fcf-7cdd-4ba0-9333-2f0b6ce7c5e2' ,

'9b33b5c6-9c33-4b0e-825a-0ed15b697fd9' ,

'744cee8a-f1d9-43ee-abe0-3bec629aad53' ,

'1bc9b580-e5d4-4e27-937b-4590507a6578' ,

'1bc9b580-e5d4-4e27-937b-4590507a6578'

)

limit 10;

---- Staff Related information old one ----

select

    ilts.learn\_training\_id as learn\_training\_id,

    ilts.version as version,

    ilts.staff\_type as staff\_type,

    SPLIT\_PART(staff\_user\_ids,

    ',',

    1) as first\_staff\_user\_id,

    SPLIT\_PART(staff\_user\_ids,

    ',',

    2) as second\_staff\_user\_id,

    SPLIT\_PART(staff\_registration\_status,

    ',',

    1) as first\_staff\_registration\_status,

    SPLIT\_PART(staff\_registration\_status,

    ',',

    2) as second\_staff\_registration\_status,\*

from

    andes.learn.ilt\_staff ilts

join andes.learn.users first\_users

on SPLIT\_PART(staff\_registration\_status,

    ',',

    1) = first\_users.id

join andes.learn.users second\_user

on SPLIT\_PART(staff\_registration\_status,

    ',',

    2) = second\_user.id

where

    ilts.learn\_training\_id = 'TPTLERN202407090021474204d7d9'

limit 10;

select

    \*

from andes.learn.training\_relationships tr

    join andes.learn.training tt on tr.target\_learn\_training\_id = tt.learn\_training\_id and tr.target\_version = tt.version

where

    source\_learn\_training\_id = 'TSELERN20240702133249dcbff9e5'

    and source\_version = '1.0';

 select

    \*

from andes.learn.training\_requisites tr

    join andes.learn.training t on tr.requisite\_id = t.learn\_training\_id

where

    tr.learn\_training\_id = 'TCRLERN202406281515580ab5627c'

    and tr.version = '1.1';

describe andes.learn.users;

--https://w.amazon.com/bin/view/BDT/Products/Andes/GettingStarted/

--https://w.amazon.com/bin/view/BDT/Support/QuickStart/PartitionandPrimaryKeysonAndes/

--Total Count - 619556836

-- Partitions - https://w.amazon.com/bin/view/BDT/Products/Andes/GettingStarted/

--userid,trainingid,lms - 506,004,913

--userid,trainingid,lms,datecreated - 615,794,244

--Training\_id - 1,191,782

--Training\_id , user\_ID - 506,024,414

-- Create a new column that maps the primary key to a fixed number of partitions(bins). For this option we are thinking about 2048 partitions.(This would work for write operation)

select count(\*) from (

select distinct lms from

andes.learn.transcripts at

where training\_id =

);

select distinct trainingid, userid from

andes.learn.transcripts at

where trainingid = 'TMTKN1T201912201218075a65f0c1'

limit 10;

select count(\*) from (select distinct trainingid, userid from

andes.learn.transcripts at

where trainingid = 'TMTKN1T201912201218075a65f0c1') at

-- Transcript Related:-

select distinct

trans.\*,

lower(users.papi\_bi\_login) as employee\_alias

,users.PAPI\_BI\_LEGACYEMPLOYEEID as employee\_id

,trans.userid

,training.default\_title as Training\_title

,trans.trainingid as learn\_training\_id

,trans.id

,trans.istoplevelflag

,trans.status

,trans.dateregistered

,trans.datecompleted

,\*

from (select \* from andes.learn.transcripts trans where trans.istoplevelflag ='true' ) trans

inner join (select \* from andes.learn.training  training

where training.default\_title in ('Driver Trainer Pre-Reconnect Knowledge Assessment'

,'Driver Trainer Post-Reconnect Knowledge Assessment')) training

on trans.trainingid = training.learn\_training\_id

inner join (select \* from andes.learn.users users where users.PAPI\_BI\_LEGACYEMPLOYEEID in (108360936,105294334)) users

on trans.userid = users.id

limit 10;

--Below ILT sample sql picked from one the users--

SELECT

    default\_title as instance\_title,

    target\_learn\_training\_id as session\_id,

    summary as instance\_details,

    training\_status as instance\_status,

    duration\_in\_iso\_8601,

    learning\_point\_of\_contact\_email,

    creation\_date,

    updated\_by as created\_by,

    comments as action\_type,

    active\_start\_date as session\_start\_date,

    active\_end\_date as session\_end\_date,

    venue\_name as location,

    catalog\_update\_date,

    registration\_min\_size,

    registration\_max\_size,

    (

        SELECT COUNT(\*)

        FROM andes\_bi.learn.transcripts

        WHERE trainingid = tr.target\_learn\_training\_id

    ) AS total\_requests,

    (

        SELECT COUNT(\*)

        FROM andes\_bi.learn.transcripts

        WHERE trainingid = tr.target\_learn\_training\_id

        AND status <> 'CANCELLED'

    ) AS seats\_taken,

    CASE

        WHEN training\_status = 'COMPLETED'

        THEN (

            SELECT COUNT(\*)

            FROM andes\_bi.learn.transcripts

            WHERE trainingid = tr.target\_learn\_training\_id

            AND status <> 'CANCELLED'

            AND status <> 'COMPLETED'

        )

        ELSE 0

    END AS noshow

FROM andes\_bi.learn.training\_relationships tr

JOIN andes\_bi.learn.training train ON tr.target\_learn\_training\_id = train.learn\_training\_id

JOIN andes\_bi.learn.ilt\_training\_attributes ita ON tr.source\_learn\_training\_id = ita.learn\_training\_id

JOIN andes\_bi.learn.training\_audit ta ON tr.source\_learn\_training\_id = ta.learn\_training\_id

WHERE

    target\_resource\_type = 'SESSION'

    AND (

        POSITION('TEVKN1T202307252230041da4e2ff' IN train.deep\_link) > 0

    )

    AND action\_type = 'Created'

    AND training\_status <> 'DRAFT';

-- Learning\_Path and Courses --

select

t.learn\_training\_id as training\_id,

ttl.training\_title as LP\_training\_title,

t.deep\_link as Learning\_path\_deep\_link,

t.training\_type as training\_type,

t.learning\_platform\_name as training\_platform\_name,

t.training\_status as training\_status,

cls.index as course\_sequence,

cls.lo\_learn\_training\_id as learn\_course\_id,

ttc.training\_title as co\_training\_title,

\*

from

andes.learn.training t ,

andes.learn.training\_title ttl ,

andes.learn.training\_title ttc,

andes.learn.course\_lp\_sequence cls

where

t.learn\_training\_id = ttl.LEARN\_TRAINING\_ID and

t.learn\_training\_id = cls.lp\_learn\_training\_id and

cls.lo\_learn\_training\_id = ttc.LEARN\_TRAINING\_ID and

t.training\_type = 'LEARNING\_PATH' and

ttl.LEARN\_TRAINING\_ID = 'LPRLERN20240430190645d9e43f42' and

ttc.version = (

select max(version)

from andes.learn.training\_title t1

where t1.learn\_training\_id = ttc.LEARN\_TRAINING\_ID

and t1.version not like '%.%.%'

)

and

ttl.version = (

select max(version)

from andes.learn.training\_title t2

where t2.learn\_training\_id = ttl.LEARN\_TRAINING\_ID

and t2.version not like '%.%.%'

)

order by cls.index;

Above LP and course has some problem.

Learn - Learning\_Path and Courses(Max LP ID, Max course ID and Max Titles):-

select

cls.lp\_learn\_training\_id as lp\_id,

cls.lo\_learn\_training\_id as learn\_course\_id,

tt.training\_title as LP\_training\_title,

t.learn\_training\_id as training\_id,

ttl.training\_title as co\_training\_title,

t.deep\_link as Learning\_path\_deep\_link,

t.training\_type as training\_type,

t.learning\_platform\_name as training\_platform\_name,

t.training\_status as training\_status,

cls.lp\_version as co\_lp\_version,

t.version as lo\_course\_version,

cls.index as course\_sequence

from

andes.learn.course\_lp\_sequence cls ,

andes.learn.training t ,

andes.learn.training\_title tt, -- Training\_Title for LP course

andes.learn.training\_title ttl -- Training\_Title for courses

where

cls.lp\_learn\_training\_id = 'LPRLERN202406172100391d036df9' and

cls.lo\_learn\_training\_id = t.learn\_training\_id and

cls.lp\_learn\_training\_id = tt.learn\_training\_id and

cls.lo\_learn\_training\_id = ttl.learn\_training\_id and

lp\_version = (

select max(lp\_version) from

andes.learn.course\_lp\_sequence cls1

where cls.lp\_learn\_training\_id = cls1.lp\_learn\_training\_id

and cls.lp\_version not like '%.%.%'

) and

t.version = (select max(t1.version) from andes.learn.training t1

where cls.lo\_learn\_training\_id = t1.learn\_training\_id

) and

ttl.version = (select max(version) from

andes.learn.training\_title title where --title.version not like '%.%.%' and

title.learn\_training\_id = ttl.learn\_training\_id) and

tt.version = (select max(version) from

andes.learn.training\_title tt1 where tt.learn\_training\_id = tt1.learn\_training\_id and

tt1.version not like '%.%.%'

);

-- KNet to Learn data comparison :- Max added on greatest registration, IsTopLevel flag is for trainings from Learn not from KNet.

WITH cte AS (

    SELECT

        SPLIT\_PART(SPLIT\_PART(t1.id, '#', 4), ':', 2) AS split\_value,

        t1.\*,

        t1.trainingId,

        u.lme  -- Ensure 'lme' is selected from the 'users' table

    FROM

        andes.learn.transcripts t1

    JOIN

        andes.learn.users u ON t1.userid = u.id

    JOIN

        andes.learn.training\_title ttl ON ttl.learn\_training\_id = t1.trainingId

    JOIN

        andes.learn.training tr ON tr.learn\_training\_id = t1.trainingId

    WHERE

        u.papi\_bi\_login IN ('shaikafq')

        AND ttl.version = tr.version

        AND ttl.version = (

            SELECT MAX(version)

            FROM andes.learn.training\_title t2

            WHERE t2.learn\_training\_id = ttl.learn\_training\_id

            AND t2.version NOT LIKE '%.%.%'

        )

        AND tr.version = (

            SELECT MAX(version)

            FROM andes.learn.training\_title t2

            WHERE t2.learn\_training\_id = tr.learn\_training\_id

            AND t2.version NOT LIKE '%.%.%'

        )

)

SELECT \*

FROM cte

WHERE (

    u.lme = 'KNet'd

    AND split\_value = (

        SELECT MAX(split\_value)

        FROM cte AS inner\_cte

        WHERE inner\_cte.trainingId = cte.trainingId

    )

)

OR u.lme <> 'KNet';

--Write Interactions/statements related queries:-

**Split\_Part Example:-**

cte ns                      AS (

    SELECT 1 AS n

    UNION

    SELECT 2

    UNION

    SELECT 3

    UNION

    SELECT 4

)

, temp\_tm2                AS (

    SELECT payee\_id

         , month\_start\_date

         , TRIM(SPLIT\_PART(territory\_id, '-', ns.n)) AS territory\_id\_split -- Some territory IDs are concated

    FROM ns

         INNER JOIN temp\_tm AS sa ON ns.n <= REGEXP\_COUNT(territory\_id, '-') + 1

    WHERE LEN(territory\_id\_split) = 18   create sample data and output for above sql

    The SQL code is using Common Table Expressions (CTEs) to parse and process data from a table. It splits a concatenated territory\_id field based on a delimiter (-), keeping entries of length 18. Here's how we can create sample data and simulate the output.

    1. Input Data

    Assume the temp\_tm table has the following data:

    payee\_id    month\_start\_date    territory\_id

    101 2024-12-01  ABCD-12345678901234-EFGH

    102 2024-12-01  XYZ-ABCDEFG-12345678

    103 2024-12-01  123456789012345678

    104 2024-12-01  ABCDEFGHIJKLMNOPQRST

    2. Steps in SQL

    1. ns CTE:

    This creates a series of numbers {1, 2, 3, 4}, used for splitting the territory\_id field.

    2. temp\_tm2 CTE:

    Joins the ns table with temp\_tm and splits territory\_id into parts using SPLIT\_PART based on the - delimiter. Filters parts where length = 18.

    3. Expected Output

    For the given input, the SQL would produce:

    payee\_id    month\_start\_date    territory\_id\_split

    101 2024-12-01  12345678901234

    103 2024-12-01  123456789012345678

    Explanation of the Output:

    Row 1 (payee\_id = 101):

    territory\_id = ABCD-12345678901234-EFGH

    Splits into {ABCD, 12345678901234, EFGH}. Only 12345678901234 has a length of 18.

    Row 2 (payee\_id = 103):

    territory\_id = 123456789012345678

    No splitting needed. This part has length 18.

    Row 3 and 4 (payee\_id = 102 and 104):

    Excluded because none of the split parts have a length of 18.

    --If we remove the condition WHERE LEN(territory\_id\_split) = 18, the SQL query will include all split parts of the territory\_id, regardless of their length.

        If we remove the condition WHERE LEN(territory\_id\_split) = 18, the SQL query will include all split parts of the territory\_id, regardless of their length.

        1. Input Data

        The same temp\_tm table is used:

        payee\_id    month\_start\_date    territory\_id

        101 2024-12-01  ABCD-12345678901234-EFGH

        102 2024-12-01  XYZ-ABCDEFG-12345678

        103 2024-12-01  123456789012345678

        104 2024-12-01  ABCDEFGHIJKLMNOPQRST

        2. Expected Output

        Without the filter, the query would produce all split parts for every territory\_id.

        payee\_id    month\_start\_date    territory\_id\_split

        101 2024-12-01  ABCD

        101 2024-12-01  12345678901234

        101 2024-12-01  EFGH

        102 2024-12-01  XYZ

        102 2024-12-01  ABCDEFG

        102 2024-12-01  12345678

        103 2024-12-01  123456789012345678

        104 2024-12-01  ABCDEFGHIJKLMNOPQRST

REGEXP\_SUBSTR(instance\_family, '[0|1|2|3|4|5|6|7|8|9]') = new\_gen input and output for this

    The SQL condition REGEXP\_SUBSTR(instance\_family, '[0|1|2|3|4|5|6|7|8|9]') = new\_gen is used to extract the first numeric character from the instance\_family string using a regular expression and compare it to the value of the column new\_gen.

    Example Explanation:-

        REGEXP\_SUBSTR Function:

        Extracts the first numeric character [0-9] from the instance\_family string.

        Returns NULL if no numeric character is found.

        Condition:

        Compares the extracted value with new\_gen.

        Only rows where the extracted numeric value matches new\_gen are included.

        Sample Input Data

        instance\_family new\_gen

        t2.medium   2

        m5.large    5

        c6a.xlarge  6

        r4.xlarge   4

        t.medium    NULL

        p3.2xlarge  3

    Expected Output:-

        For the given condition:

        instance\_family new\_gen

        t2.medium   2

        m5.large    5

        c6a.xlarge  6

        r4.xlarge   4

        p3.2xlarge  3

How It Works

Row 1: instance\_family = 't2.medium'

REGEXP\_SUBSTR extracts 2.

2 = new\_gen → Included.

Row 2: instance\_family = 'm5.large'

REGEXP\_SUBSTR extracts 5.

5 = new\_gen → Included.

Row 3: instance\_family = 'c6a.xlarge'

REGEXP\_SUBSTR extracts 6.

6 = new\_gen → Included.

Row 4: instance\_family = 'r4.xlarge'

REGEXP\_SUBSTR extracts 4.

4 = new\_gen → Included.

Row 5: instance\_family = 't.medium'

No numeric character, REGEXP\_SUBSTR returns NULL.

NULL != new\_gen → Excluded.

Row 6: instance\_family = 'p3.2xlarge'

REGEXP\_SUBSTR extracts 3.

3 = new\_gen → Included.

**NEJ:-**

with section\_range as (

    select ac.curriculum\_id , ac.parent\_training\_id , ac.training\_id , att.training\_title as training\_title  , ac.training\_display\_sequence ,

     LEAD(training\_display\_sequence , 1) OVER(

        ORDER BY training\_display\_sequence

      ) AS next\_Section\_id,

       lag(training\_display\_sequence , 1) OVER(

        ORDER BY training\_display\_sequence

      ) AS prev\_section\_id,

      ac.al\_lms

      from

    "knet-data".alrn\_curriculum ac,

    "knet-data".alrn\_training\_title att

    where --ac.curriculum\_id in ('028a2e92-108f-4390-84c4-3e1d27446c5e') and

    ac.curriculum\_id in ('028a2e92-108f-4390-84c4-3e1d27446c5e') and

    ac.required\_training\_per\_section is not null and --for section\_title

    att.training\_id = ac.training\_id and

    att.is\_default\_language = 'True' and

    att.al\_lms = ac.al\_lms

    order by training\_display\_sequence

    ),

    curr as(

    select  ac.curriculum\_id , ac.parent\_training\_id , ac.training\_id , att.training\_title as training\_title\_t  , ac.training\_display\_sequence

    , lag(training\_display\_sequence , 1) OVER(

        ORDER BY training\_display\_sequence

      ) AS previous\_section\_id  ,att.training\_title ,

      case when ac.required\_training\_per\_section is null then 'Training' else 'Section\_Title' end as curriculum\_section\_part ,

      ac.al\_lms

      from

    "knet-data".alrn\_curriculum ac,

    "knet-data".alrn\_training\_title att

    where ac.curriculum\_id in ('028a2e92-108f-4390-84c4-3e1d27446c5e') and

    --ac.required\_training\_per\_section is null and

    att.training\_id = ac.training\_id and

    att.al\_lms = ac.al\_lms and

    att.is\_default\_language = 'True'

    order by training\_display\_sequence

    )

    select

    sr.curriculum\_id,

    att.training\_title as curriculum\_title,

    c.parent\_training\_id,

    sr.training\_title as section\_title,

    c.training\_id as section\_training\_id,

    at2.training\_type\_description ,

    c.training\_title\_t as trainings\_under\_section

    from

    section\_range sr,

    curr c ,

    "knet-data".alrn\_training at2 ,

    "knet-data".alrn\_training\_title att

    where (

       (c.training\_display\_sequence between sr.training\_display\_sequence and sr.next\_Section\_id  )

    or (sr.next\_section\_id is null and c.training\_display\_sequence > sr.training\_display\_sequence)

    or (sr.prev\_section\_id is null and sr.training\_display\_sequence <> 1 and c.training\_display\_sequence < sr.training\_display\_sequence)

    )

    and

    c.curriculum\_section\_part <> 'Section\_Title' and

    c.training\_id = at2.training\_id and

    c.al\_lms = at2.al\_lms and

    att.training\_id = sr.curriculum\_id and

    att.al\_lms = sr.al\_lms

    order by c.training\_display\_sequence asc, sr.training\_display\_sequence asc;

**Evaluations:-**

SELECT alrn\_evaluation\_title.assessment\_type,

       alrn\_evaluation\_text.text AS "Question",

       alrn\_evaluation\_question.question\_id,

       alrn\_evaluation\_question\_type.question\_type AS "Question Type",

       alrn\_evaluation\_base.csod\_user\_id,

       alrn\_evaluation\_base.eval\_submitted\_date,

       alrn\_evaluation\_base.training\_id ,

       alrn\_evaluation\_base.container\_id ,

       alrn\_evaluation\_text.text\_id,

       alrn\_evaluation\_base.eval\_submitted\_date ,

       alrn\_evaluation\_base.csod\_eval\_session\_id,

       COALESCE(et2.text,eresp.response\_text) AS answer

FROM "knet-data".alrn\_evaluation\_text,

     "knet-data".alrn\_evaluation\_base,

     "knet-data".alrn\_evaluation\_structure,

     "knet-data".alrn\_evaluation\_question,

     "knet-data".alrn\_evaluation\_question\_type,

     "knet-data".alrn\_evaluation\_title ,

     "knet-data".alrn\_evaluation\_result aer,

     "knet-data".alrn\_evaluation\_response eresp,

     "knet-data".alrn\_evaluation\_answer eans,

     "knet-data".alrn\_evaluation\_text et2

WHERE alrn\_evaluation\_base.csod\_eval\_session\_id = aer.csod\_eval\_session\_id

and alrn\_evaluation\_base.al\_lms=aer.al\_lms

and aer.question\_instance\_id=alrn\_evaluation\_structure.question\_instance\_id

and aer.al\_lms=alrn\_evaluation\_structure.al\_lms

AND   alrn\_evaluation\_structure.question\_text\_id = alrn\_evaluation\_question.question\_id

and alrn\_evaluation\_structure.al\_lms=alrn\_evaluation\_question.al\_lms

and    alrn\_evaluation\_question.question\_type\_id = alrn\_evaluation\_question\_type.question\_type\_id

and  alrn\_evaluation\_question.al\_lms = alrn\_evaluation\_question\_type.al\_lms

and   alrn\_evaluation\_question\_type.is\_default\_language = 'True'

AND   alrn\_evaluation\_question.question\_text\_id = alrn\_evaluation\_text.text\_id

and alrn\_evaluation\_question.al\_lms=alrn\_evaluation\_text.al\_lms

and   alrn\_evaluation\_title.container\_id = alrn\_evaluation\_base.container\_id

and   alrn\_evaluation\_title.al\_lms=alrn\_evaluation\_base.al\_lms

and   aer.assessment\_result\_id=eresp.assessment\_result\_id

and   aer.al\_lms=eresp.al\_lms

and   eresp.answer\_item\_id=eans.answer\_item\_id

and   eresp.al\_lms=eans.al\_lms

and   aer.assessment\_result\_id=eresp.assessment\_result\_id

and   aer.al\_lms=eresp.al\_lms

and   eans.answer\_text\_id=et2.text\_id and lower(et2.is\_default\_language) = 'true'

and   eans.al\_lms=et2.al\_lms

and   alrn\_evaluation\_base.training\_id in ('879da9c3-af17-4159-b108-c08678751e53')

AND   alrn\_evaluation\_structure.al\_lms = alrn\_evaluation\_question.al\_lms

AND   alrn\_evaluation\_question.al\_lms =  alrn\_evaluation\_question\_type.al\_lms

AND   alrn\_evaluation\_base.eval\_submitted\_date >= '2023-08-01'

and   alrn\_evaluation\_base.csod\_user\_id in ('5968024');

--and  alrn\_evaluation\_base.csod\_user\_id='326680'

--and  alrn\_evaluation\_question\_type.question\_type in ('Yes/No')

ebase - instances of evaluation on the KNet Training Assessment  as taken by users

eresult - ebase

question evaluationstructure

question

text for question

eresult - ebase

response

answer

text for answer

--83884002

select \* from "knet-data".alrn\_evaluation\_result aer -- stored questions and result instance id's

where aer.csod\_eval\_session\_id = '83884002'--811502443

select \* from "knet-data".alrn\_evaluation\_response aer

where aer.assessment\_result\_id = '811502443'

select \* from "knet-data".alrn\_evaluation\_answer aea

where aea.answer\_item\_id in ('4388267','4393174');

select \* from "knet-data".alrn\_evaluation\_text aet

where aet.text\_id  in ('4383360')

select \* from "knet-data".alrn\_evaluation\_structure aes

where aes.question\_instance\_id in ('6266645')

SELECT \*

FROM "knet-data".alrn\_evaluation\_base,

     "knet-data".alrn\_evaluation\_result,

     "knet-data".alrn\_evaluation\_response,

     "knet-data".alrn\_evaluation\_text,

     "knet-data".alrn\_evaluation\_answer

WHERE alrn\_evaluation\_base.csod\_eval\_session\_id = alrn\_evaluation\_result.csod\_eval\_session\_id

AND   alrn\_evaluation\_result.assessment\_result\_id = alrn\_evaluation\_response.assessment\_result\_id

AND   alrn\_evaluation\_response.answer\_item\_id = alrn\_evaluation\_text.text\_id limit 10;

  left join knet-evaluation-response eresp on eresult.assessment\_result\_id=eresp.assessment\_result\_id

        left join knet-evaluation-answer eans on eresp.answer\_item\_id=eans.answer\_item\_id

and   alrn\_evaluation\_base.training\_id in ('879da9c3-af17-4159-b108-c08678751e53')

**Evaluations sections:-**

--- This is for section heading , questions and sub-questions

SELECT distinct

  aestr.al\_lms,

  aestr.container\_id,

  aet.text AS "Evaluation\_name",

  aetst.text AS  "Section\_heading",

  aet2.text as "Parent\_questions",

  aetrq.text AS "Subquestions",

  aeaet.text AS "Answers\_text",

  alrn\_evaluation\_question\_type.question\_type AS "Question Type",

  et.eval\_type,

  et.assessment\_type,

  aestr.container\_id,

  et.is\_active,

  aeq.question\_id,

  aestr.question\_text\_id,

  aets.section\_text\_id,

  aestr.question\_instance\_id,

  aestr.parent\_question\_instance\_id,

  aestr.parent\_question\_text\_id

from

-------------------evaluation\_type,assessmen\_type,is\_active---------------------------------

  "knet-data".alrn\_evaluation\_title et

left JOIN

  "knet-data".alrn\_evaluation\_structure aestr

  ON aestr.container\_id = et.container\_id AND aestr.al\_lms = et.al\_lms

left join

----------------- question\_id / Answers\_available for Likert Scale question type -----------------

  "knet-data".alrn\_evaluation\_question aeq

  ON aeq.question\_id = aestr.question\_text\_id AND aeq.al\_lms = aestr.al\_lms

left JOIN

  "knet-data".alrn\_evaluation\_text aetrq

  ON aetrq.text\_id = aeq.question\_text\_id AND aetrq.al\_lms = aeq.al\_lms

------     Section text Information for section text  -----------------------

left JOIN

  "knet-data".alrn\_evaluation\_section\_text aets

  ON aets.section\_text\_id = aestr.section\_text\_id AND aets.al\_lms = aestr.al\_lms

left JOIN

  "knet-data".alrn\_evaluation\_text aetst

  ON aetst.text\_id = aets.text\_id AND aetst.al\_lms = aets.al\_lms

------------------------------ parent\_text within the section -------------------

left join

    "knet-data".alrn\_evaluation\_text aet2

  on aet2.text\_id = aestr.parent\_question\_text\_id

--------- Correct title   -----------------------

left JOIN

  "knet-data".alrn\_evaluation\_text aet

  ON aet.text\_id = et.eval\_title\_id AND aet.al\_lms = et.al\_lms

------------------------ answer chosen for given question not valid in this case ------------------

left JOIN

  "knet-data".alrn\_answer\_bank aab

  on aab.question\_id = aeq.question\_id and aab.al\_lms = aeq.al\_lms

left JOIN

  "knet-data".alrn\_evaluation\_answer aea

  ON aea.answer\_item\_id = aab.answer\_id and aea.al\_lms = aab.al\_lms

left JOIN

  "knet-data".alrn\_evaluation\_text aeaet

  ON aea.answer\_text\_id = aeaet.text\_id and aea.al\_lms = aeaet.al\_lms

----------------------------------- question\_type ----------------------------------

left join

  "knet-data".alrn\_evaluation\_question\_type alrn\_evaluation\_question\_type

  on aeq.question\_type\_id = alrn\_evaluation\_question\_type.question\_type\_id and  aeq.al\_lms = alrn\_evaluation\_question\_type.al\_lms

------------------------------------------------------------------------------------------------------------------------------------

where

  --aestr.container\_id = 49516

    aet2.al\_lms = aestr.al\_lms

  AND aestr.al\_lms = 'KNet'

  AND aet.text IS NOT NULL

  AND aetst.text IS NOT NULL

  AND aetrq.text IS NOT null

  and aetrq.language\_id = 1

  and lower(alrn\_evaluation\_question\_type.is\_default\_language) = 'true'

  and aet.text in ('EU Ops LD-Emerging Leaders Conference\_2014\_v1')

  --and alrn\_evaluation\_question\_type.question\_type in ('Likert Scale')

  --and aetst.text in ('Session Feedback')

  and aet2.is\_default\_language = 'True'

 order by aetst.text desc;

**Recursive CTE Example:-**

select \* from job\_positions;

select \* from job\_employees;

with cte as (

select id, name, position\_id, row\_number() over (partition by position\_id order by position\_id) as rn from job\_employees

)

select \* from cte order by 1;

**Recursive CTE example:-**

with recursive temp\_tbl(id,title,groups,levels,payscale,totalpost,rn) as (

    select id,title,groups,levels,payscale,totalpost, 1 as rn from job\_positions

    union all

    select id,title,groups,levels,payscale,totalpost, rn+1 as rn from temp\_tbl

    where  rn+1 <= totalpost

    )

    select \* from temp\_tbl order by 1;

SELECT

    a.training\_id,

    CASE

        WHEN training\_id = '3b0e1c9a-344f-4232-b2b2-dbc2ea89c938'

        THEN 'New Training'

        ELSE 'Old Training'

    END AS training\_type, -- renamed column alias

    CASE

        WHEN transcript\_status\_id = 1073741888 THEN 'Completed (Equivalent)'

        WHEN transcript\_status\_id = 16 THEN 'Registered'

        WHEN transcript\_status\_id = 16777216 THEN 'Old Version'

        WHEN transcript\_status\_id = 32 THEN 'In Progress'

        WHEN transcript\_status\_id = 34359738432 THEN 'Exempt'

        WHEN transcript\_status\_id = 589824 THEN 'Pending Prerequisite / Past Due'

        WHEN transcript\_status\_id =  524288 THEN 'Pending Prerequisite'

        WHEN transcript\_status\_id =  64 THEN 'Completed'

        Else 'Registered / Past Due'

    END AS Transcript\_Status,

    a.csod\_user\_id AS user\_id,

    a.CSOD\_ASSIGNOR\_ID AS assign\_id,

    a.transcript\_reg\_date AS reg\_date,

    a.transcript\_assignment\_date,

    a.transcript\_start\_date,

    a.transcript\_completion\_date,

    a.transcript\_status\_id,

    a.is\_success,

    a.is\_standalone,

    a.is\_assigned,

    b.amzn\_alias,

    a.TRANSCRIPT\_DELIVERY\_METHOD\_ID,

    a.IS\_GREATEST\_REG\_NUM,

    a.IS\_LATEST\_TRAINING\_VERSION,

    b.account\_status,

    b.CSOD\_MANAGER\_ID

FROM "knet-data".alrn\_transcript a

inner join "knet-data".alrn\_user b

on a.csod\_user\_id = b.csod\_user\_id

WHERE training\_id IN ('3b0e1c9a-344f-4232-b2b2-dbc2ea89c938', '4d7c3b23-4087-463c-8e44-c21b6f0718a4')

and b.al\_lms = 'KNet'

and b.account\_status = 1

--------------------------------------

-- find courses by KNET IDs

SELECT DISTINCT  kt.csod\_user\_id, tp.training\_title, kt.al\_lms,

                kt.transcript\_status\_id,

                kt.transcript\_reg\_date,

                kt.transcript\_completion\_date,

                kt.transcript\_due\_date

FROM knet\_transform.alrn\_transcript kt

    INNER JOIN (

        SELECT t.training\_id, ktt.training\_title

        FROM knet\_transform.alrn\_training t

            INNER JOIN knet\_transform.alrn\_training\_title ktt ON (t.training\_id = ktt.training\_id)

        WHERE ktt.training\_title like '%Not About the Badge%'

    ) tp on tp.training\_id=kt.training\_id

WHERE kt.csod\_user\_id in  (10188020, 5230871)

------------------------------------------------------

select csod\_user\_id,

                         training\_id,

                         at.transcript\_status\_id,

                         at.al\_lms,

                         status\_title,

                         transcript\_start\_date,

                         transcript\_completion\_date,

                         transcript\_last\_action\_date,

                         transcript\_due\_date,

                         transcript\_reg\_date

                  from "knet-data".alrn\_transcript at

                           left join "knet-data".alrn\_transcript\_status\_language atsl

                                     on at.transcript\_status\_id = atsl.transcript\_status\_id and at.al\_lms = atsl.al\_lms

                  where training\_id in (select distinct training\_id

                                        from "knet-data".alrn\_training\_title

                                        where training\_title =

                                              'AMZN: Food Safety: Food Information to Consumers (FIC) Program')

--   and transcript\_completion\_date is null

                    and atsl.language\_id = 1

--------------------------------------

We are not seeing below list of users on the following course: Cascading Education - Psychological Safety

(kt.training\_id = 'f3d9f848-093a-496e-84c5-6bd7e172b403')

csod\_user\_id,employee\_login

1025196,rosscamp --

10100772,rosscamp

10669560,giafonzo --

13241726,giafonzo

16815495,basharfa

12804838 --> KNet

Query used -

SELECT DISTINCT kt.csod\_user\_id, kt.al\_lms, kt.training\_id,

kt.transcript\_status\_id,

kt.transcript\_reg\_date,

kt.transcript\_completion\_date,

kt.transcript\_due\_date,

kt.is\_archived,

kt.is\_removed,

kt.is\_latest\_training\_version,

kt.load\_date

FROM alrn\_transcript kt

WHERE kt.csod\_user\_id in (1025196,10100772,10669560,12804838,13241726,16815495)

and kt.training\_id = 'f3d9f848-093a-496e-84c5-6bd7e172b403'

------------------------------------------------------------------------------------------------------------

Checking if they are valid Csod\_User:-

SELECT DISTINCT

kt.csod\_user\_id,

kt.al\_lms,

kt.training\_id,

kt.transcript\_status\_id,

kt.transcript\_reg\_date,

kt.transcript\_completion\_date,

kt.transcript\_due\_date,

kt.is\_archived,

kt.is\_removed,

kt.is\_latest\_training\_version--,

--kt.load\_date

FROM  "knet-data".alrn\_transcript kt

WHERE kt.csod\_user\_id in (1025196,10100772,10669560,12804838,13241726,16815495)

and kt.training\_id = 'f3d9f848-093a-496e-84c5-6bd7e172b403';

select al\_lms,au.\*  from "knet-data".alrn\_user au

WHERE csod\_user\_id in (16815495,12804838) /\*and amzn\_empl\_id in ('112281977')\*/ order by 1;

------------------------------------------------------------------------------------------------------------

We generate a monthly report that shows which users who are assigned to RDA KNET training vs how many users completed it every month.

The report shows different assignment id's of the department completing the training.

I am interested i knowing which KNET table holds that information of departments as well

I am not able to match the numbers of total people assigned vs completed with the dashboard I created using the Knet alms table

------------------------------------------------------------------------------------------------------------

select etest.\*

from "knet-data".alrn\_evaluation\_test as etest

inner join "knet-data".alrn\_user as au

on etest.csod\_user\_id = au.csod\_user\_id

where etest.csod\_user\_id in (

    12932875,

    16934390,

    102265049 --Don't know why the Excel report shows this id for

)

and etest.training\_id = '5c023802-dca2-4b1f-9fb7-524d7ea5cf62'

limit 100;

--------------------

----------------------------------------------------------------------------------------------------------------------

Basics

A user will now have multiple identities one for each portal(KNet and KNet2) in the user table. If you want to find consolidated transcript please

drive your queries based on amzn\_empl\_id otherwise use al\_lms as a filter to find portal specific records.

1. User Name:

SELECT amzn\_alias FROM alrn\_user, alrn\_transcript WHERE alrn\_user.csod\_user\_id = alrn\_transcript.csod\_user\_id

AND alrn\_user.al\_lms = alrn\_user.al\_lms AND al\_lms = 'KNet'

2. Training Title:

SELECT training\_title FROM alrn\_training\_title, alrn\_transcript

WHERE alrn\_transcript.training\_id = alrn\_training-title.training\_id

AND alrn\_training\_title.al\_lms = alrn\_transcript.al\_lms AND al\_lms = 'KNet'

3. Training Start Date:

SELECT transcript\_start\_date FROM alrn\_transcript WHERE training\_id = "<Insert Training ID>" AND al\_lms = 'KNet'

4. Training ID: You should ideally have this to start with. Alternatively, look for your training in training-title or training-subject tables

5. Location:

----------------------------------------------------------------------------------------------------------------------

SELECT location\_building\_code

FROM employee,

     alrn\_transcript,

     alrn\_user

WHERE alrn\_transcript.csod\_user\_id = alrn\_user.csod\_user\_id,

AND   alrn\_user.amzn\_empl\_id = employee.emplid

AND   alrn\_transcript.al\_lms = alrn\_user.al\_lms

----------------------------------------------------------------------------------------------------------------------

Evaluation Related Fields

1. Evaluation Name & Evaluation Type:

SELECT alrn\_evaluation\_text.text AS "Evaluation Name",

       alrn\_evaluation\_title.eval\_type AS "Evaluation Type"

FROM alrn\_evaluation\_base,

     alrn\_evaluation\_title,

     alrn\_evaluation\_text

WHERE alrn\_evaluation\_base.container\_id = alrn\_evaluation\_title.container\_id

AND   alrn\_evaluation\_title.eval\_title\_id = alrn\_evaluation\_text.text\_id

AND   alrn\_evaluation\_base.al\_lms = alrn\_evaluation\_title.al\_lms

AND   alrn\_evaluation\_title.al\_lms = alrn\_evaluation\_text.al\_lms

----------------------------------------------------------------------------------------------------------------------

2. Date Submitted: alrn\_evaluation\_base.eval\_submitted\_date -- (datetime in UTC!)

----------------------------------------------------------------------------------------------------------------------

3. Question & Question Type:

SELECT alrn\_evaluation\_text.text AS "Question",

       alrn\_evaluation\_question\_type.question\_type AS "Question Type"

FROM alrn\_evaluation\_text,

     alrn\_evaluation\_base,

     alrn\_evaluation\_structure,

     alrn\_evaluation\_question,

     alrn\_evaluation\_question\_type

WHERE alrn\_evaluation\_base.container\_id = alrn\_evaluation\_structure.container\_id

AND   alrn\_evaluation\_structure.question\_text\_id = alrn\_evaluation\_question.question\_id

AND   alrn\_evaluation\_question.question\_type\_id = alrn\_evaluation\_question\_type.question\_type\_id

AND   alrn\_evaluation\_question.question\_text\_id = alrn\_evaluation\_text.text\_id

AND   alrn\_evaluation\_text.al\_lms = alrn\_evaluation\_base.al\_lms

AND   alrn\_evaluation\_base.al\_lms = alrn\_evaluation\_structure.al\_lms

AND   alrn\_alrn\_evaluation\_structure.al\_lms = alrn\_evaluation\_question.al\_lms

AND   alrn\_evaluation\_question.al\_lms = alrn\_evaluation\_question\_type

----------------------------------------------------------------------------------------------------------------------

4.1 Question User Response (if Question Type = Text Only)

SELECT alrn\_evaluation\_response.response\_text AS "Question User Response (Text)"

FROM alrn\_evaluation\_base,

     alrn\_evaluation\_result,

     alrn\_evaluation\_response

WHERE alrn\_evaluation\_base.csod\_eval\_session\_id = alrn\_evaluation\_result.csod\_eval\_session\_id

AND   alrn\_evaluation\_result.assessment\_result\_id = alrn\_evaluation\_response.assessment\_result\_id

AND   alrn\_evaluation\_base.al\_lms = alrn\_evaluation\_result.al\_lms

AND   alrn\_evaluation\_result.al\_lms = alrn\_evaluation\_response.al\_lms

4.2 Question User Response (if Question Type = Yes/No or Multiple Choice)

SELECT alrn\_evaluation\_text.text AS "Question User Response (Multiple Choice)"

 FROM alrn\_evaluation\_base,

      alrn\_evaluation\_result,

      alrn\_evaluation\_response,

      alrn\_evaluation\_text

WHERE alrn\_evaluation\_base.session\_id = alrn\_evaluation\_result.csod\_eval\_session\_id

AND   alrn\_evaluation\_result.assessment\_result\_id = alrn\_evaluation\_response.assessment\_result\_id

AND   alrn\_evaluation\_response.answer\_item\_id = alrn\_evaluation\_text.text\_id

AND   alrn\_evaluation\_base.al\_lms = alrn\_evaluation\_result.al\_lms

AND   alrn\_evaluation\_result.al\_lms = alrn\_evaluation\_response

AND alrn\_evaluation\_response.al\_lms = alrn\_evaluation\_text.al\_lms

----------------------------------------------------------------------------------------------------------------------

/\*I have a confidential pending legal matter requiring all training given to Melissa White (alias: whtmeli; EID: 107501012) from January 1, 2021 to present regarding her position as a Senior Human Resources Assistant.  I must include a description of the training (e.g. whether it was in

Person /Computer-based,

The date

Place of training -- Not found

The name of the individual(s) who conducted the training

Those who attended the training - had to write cte for this

The subject matter covered in the training --subject

Responsibilities of an employee - Added Business\_title

SIM\_Number:- https://t.corp.amazon.com/V933429835/overview

-------------------------------------------------------------------------------------

\*/

with melissa\_details as (

    select  distinct

    ttl.training\_title,

    tr.training\_id,

    tr.transcript\_completion\_date,

    tra.training\_type\_description,

    --tra.is\_virtual\_training,

    tra.training\_type,

    tra.al\_lms,

    tra.is\_mobile\_compatible,

    tr.transcript\_start\_date,

    au.business\_title,

    au.User\_first\_name,

    au.User\_last\_name,

    au2.Trainer\_first\_name,

    au2.Trainer\_last\_name

    from

        (select training\_id ,transcript\_start\_date,transcript\_completion\_date, al\_lms, csod\_user\_id

        from "knet-data".alrn\_transcript where is\_latest\_training\_version = 'True' and al\_lms in ('KNet','KNet2')) tr

        inner join

        (select training\_title,training\_id,al\_lms from "knet-data".alrn\_training\_title /\*where ttl.training\_title in ('[English] 2023 Recharge: Sustainability Training') and al\_lms='KNet'\*/) ttl

        on tr.training\_id= ttl.training\_id and  tr.al\_lms = ttl.al\_lms

        inner join

        (select training\_type,training\_created\_by\_user\_id

        ,is\_virtual\_training,training\_type\_description,is\_mobile\_compatible, training\_id, al\_lms

        from "knet-data".alrn\_training tra where tra.al\_lms in ('KNet','KNet2')) tra

        on tr.training\_id= tra.training\_id and tr.al\_lms = tra.al\_lms

        inner join

        (select business\_title, name\_first as User\_first\_name, name\_last as User\_last\_name , csod\_user\_id, al\_lms

        from "knet-data".alrn\_user au where lower(au.amzn\_alias) like '%whtmeli%' and al\_lms in ('KNet','KNet2')) au

        on au.csod\_user\_id=tr.csod\_user\_id and au.al\_lms = tr.al\_lms -- 250

        inner join

        (select name\_first as Trainer\_first\_name,name\_last as Trainer\_last\_name , csod\_user\_id

        from "knet-data".alrn\_user au where al\_lms in ('KNet','KNet2')) au2

        on au2.csod\_user\_id=tra.training\_created\_by\_user\_id

        left join

        (select subject\_title,training\_id from "knet-data".alrn\_training\_subject ) ts

        on ts.training\_id=tr.training\_id

        order by 1, 2),--250/18sec

    melissa\_team as (select  distinct

    ttl.training\_title,

    tr.training\_id,

    tr.transcript\_completion\_date,

    tra.training\_type\_description,

    --tra.is\_virtual\_training,

    tra.training\_type,

    tra.al\_lms,

    tra.is\_mobile\_compatible,

    tr.transcript\_start\_date,

    au.business\_title,

    au.User\_first\_name,

    au.User\_last\_name,

    au2.Trainer\_first\_name,

    au2.Trainer\_last\_name

    from

        (select training\_id ,transcript\_start\_date,transcript\_completion\_date, al\_lms, csod\_user\_id

        from "knet-data".alrn\_transcript where is\_latest\_training\_version = 'True' and al\_lms in ('KNet','KNet2')) tr

        inner join

        (select training\_title,training\_id,al\_lms from "knet-data".alrn\_training\_title /\*where ttl.training\_title in ('[English] 2023 Recharge: Sustainability Training') and al\_lms='KNet'\*/) ttl

        on tr.training\_id= ttl.training\_id and  tr.al\_lms = ttl.al\_lms

        inner join

        (select training\_type,training\_created\_by\_user\_id,is\_virtual\_training,training\_type\_description,is\_mobile\_compatible, training\_id, al\_lms

        from "knet-data".alrn\_training tra where tra.al\_lms in ('KNet','KNet2') and training\_id in (select distinct training\_id from melissa\_details m

        where training\_type\_description in ('Online Class'))) tra

        on tr.training\_id= tra.training\_id and tr.al\_lms = tra.al\_lms inner join

        (select business\_title, name\_first as User\_first\_name, name\_last as User\_last\_name , csod\_user\_id, al\_lms

        from "knet-data".alrn\_user au where /\*lower(au.amzn\_alias) like '%whtmeli%' and \*/al\_lms in ('KNet','KNet2')) au

        on au.csod\_user\_id=tr.csod\_user\_id and au.al\_lms = tr.al\_lms -- 250

        inner join

        (select name\_first as Trainer\_first\_name,name\_last as Trainer\_last\_name , csod\_user\_id

        from "knet-data".alrn\_user au where al\_lms in ('KNet','KNet2')) au2

        on au2.csod\_user\_id=tra.training\_created\_by\_user\_id

        left join

        (select subject\_title,training\_id from "knet-data".alrn\_training\_subject ) ts

        on ts.training\_id=tr.training\_id

        order by 1, 2)

    select count(\*) from (

    select distinct \* from melissa\_details

    union

    select distinct \* from melissa\_team) ab -- 2m/200 rows/ Melissa entries total 250/14316366 those who attended training

SQL :-

Statement:-

    COALESCE(num\_comp\_plan\_others, LAG(num\_comp\_plan\_others)

    IGNORE NULLS OVER (PARTITION BY sfdc\_customer\_id,fiscal\_year ORDER BY quarter\_begin\_date )) AS num\_comp\_plan\_others

    Details:- This SQL snippet uses the COALESCE function combined with a LAG window function to fill in missing (NULL) values in the num\_comp\_plan\_others column.

    Here's an explanation of its components and potential use case:

    Key Components

    1. COALESCE:

    Returns the first non-NULL value from its arguments.

    In this case, it checks:

    The current row's value of num\_comp\_plan\_others.

    The value of num\_comp\_plan\_others from the previous non-NULL row in the same partition.

    2. LAG Function:

    Retrieves the value of num\_comp\_plan\_others from the previous row in the defined window (partition and order).

    IGNORE NULLS ensures it skips over NULL values when finding the previous row.

    3. OVER Clause:

    PARTITION BY sfdc\_customer\_id, fiscal\_year: Groups data by customer and fiscal year.

    ORDER BY quarter\_begin\_date: Processes rows in chronological order within each partition.

    4. Result:

    If the current row's num\_comp\_plan\_others is NULL, it fills it with the most recent non-NULL value from the same partition.

    Example Use Case

    To fill gaps in a time series dataset where num\_comp\_plan\_others may be missing (e.g., compensation plans for sales teams), ensuring continuity by backfilling with the last known value.

    Example Use Case

    To fill gaps in a time series dataset where num\_comp\_plan\_others may be missing (e.g., compensation plans for sales teams),

    ensuring continuity by backfilling with the last known value.

    Input Data

    sfdc\_customer\_id    fiscal\_year quarter\_begin\_date  num\_comp\_plan\_others

    101 2024    2024-01-01  5

    101 2024    2024-04-01  NULL

    101 2024    2024-07-01  8

    102 2024    2024-01-01  NULL

    102 2024    2024-04-01  3

    102 2024    2024-07-01  NULL

    Expected Output

    sfdc\_customer\_id    fiscal\_year quarter\_begin\_date  num\_comp\_plan\_others

    101 2024    2024-01-01  5

    101 2024    2024-04-01  5

    101 2024    2024-07-01  8

    102 2024    2024-01-01  NULL

    102 2024    2024-04-01  3

    102 2024    2024-07-01  3

Python:-

    import pandas as pd

    # Sample DataFrame

    data = {

        'sfdc\_customer\_id': [101, 101, 101, 102, 102, 102],

        'fiscal\_year': [2024, 2024, 2024, 2024, 2024, 2024],

        'quarter\_begin\_date': ['2024-01-01', '2024-04-01', '2024-07-01', '2024-01-01', '2024-04-01', '2024-07-01'],

        'num\_comp\_plan\_others': [5, None, 8, None, 3, None]

    }

    df = pd.DataFrame(data)

    # Convert 'quarter\_begin\_date' to datetime

    df['quarter\_begin\_date'] = pd.to\_datetime(df['quarter\_begin\_date'])

    # Sort the DataFrame by 'sfdc\_customer\_id', 'fiscal\_year', and 'quarter\_begin\_date'

    df = df.sort\_values(by=['sfdc\_customer\_id', 'fiscal\_year', 'quarter\_begin\_date'])

    # Backfill missing values within each group

    df['num\_comp\_plan\_others\_filled'] = df.groupby(['sfdc\_customer\_id', 'fiscal\_year'])['num\_comp\_plan\_others'].apply(lambda x: x.ffill())

    # Display the result

    print(df)

    Output

    sfdc\_customer\_id    fiscal\_year quarter\_begin\_date  num\_comp\_plan\_others    num\_comp\_plan\_others\_filled

    101 2024    2024-01-01  5.0 5.0

    101 2024    2024-04-01  NaN 5.0

    101 2024    2024-07-01  8.0 8.0

    102 2024    2024-01-01  NaN NaN

    102 2024    2024-04-01  3.0 3.0

    102 2024    2024-07-01  NaN 3.0

In Python, you can replicate this functionality using the pandas library. Here's how you can achieve the same result:

Code Example

python

Copy code

import pandas as pd

# Sample DataFrame

data = {

    'sfdc\_customer\_id': [101, 101, 101, 102, 102, 102],

    'fiscal\_year': [2024, 2024, 2024, 2024, 2024, 2024],

    'quarter\_begin\_date': ['2024-01-01', '2024-04-01', '2024-07-01', '2024-01-01', '2024-04-01', '2024-07-01'],

    'num\_comp\_plan\_others': [5, None, 8, None, 3, None]

}

df = pd.DataFrame(data)

# Convert 'quarter\_begin\_date' to datetime

df['quarter\_begin\_date'] = pd.to\_datetime(df['quarter\_begin\_date'])

# Sort the DataFrame by 'sfdc\_customer\_id', 'fiscal\_year', and 'quarter\_begin\_date'

df = df.sort\_values(by=['sfdc\_customer\_id', 'fiscal\_year', 'quarter\_begin\_date'])

# Backfill missing values within each group

df['num\_comp\_plan\_others\_filled'] = df.groupby(['sfdc\_customer\_id', 'fiscal\_year'])['num\_comp\_plan\_others'].apply(lambda x: x.ffill())

# Display the result

print(df)

Output

sfdc\_customer\_id    fiscal\_year quarter\_begin\_date  num\_comp\_plan\_others    num\_comp\_plan\_others\_filled

101 2024    2024-01-01  5.0 5.0

101 2024    2024-04-01  NaN 5.0

101 2024    2024-07-01  8.0 8.0

102 2024    2024-01-01  NaN NaN

102 2024    2024-04-01  3.0 3.0

102 2024    2024-07-01  NaN 3.0

Explanation

Data Sorting:

The data is sorted by sfdc\_customer\_id, fiscal\_year, and quarter\_begin\_date to ensure proper ordering within groups.

Group Processing:

The groupby function groups data by sfdc\_customer\_id and fiscal\_year.

Forward Fill:

The ffill method within apply fills NaN values with the most recent non-NaN value within each group, similar to the SQL logic with COALESCE and LAG IGNORE NULLS.